

IN THE UNITED STATES DISTRICT COURT
FOR THE NORTHERN DISTRICT OF CALIFORNIA

ORACLE AMERICA, INC.,

No. C 10-03561 WHA

Plaintiff,

v.

GOOGLE INC.,

Defendant.

**ORDER OVERRULING SPOON-
FEEDING OBJECTION TO
KEMERER EXPERT
TESTIMONY**

This order answers the question left open in the earlier order concerning Dr. Chris F. Kemerer (Dkt. No. 1806).

*

*

*

Google raises five challenges to Dr. Chris F. Kemerer's methodology following the recent deposition of Rohit Chatterjee, who collaborated with Kemerer in performing his "stability" and "centrality" analyses in his expert report. *First*, Kemerer worked with Chatterjee and the rest of the Keystone research team for the first time to prepare his expert report in this case. *Second*, Kemerer did not attach the Python scripts used by the Keystone team to his report. *Third*, Kemerer decided which tests to use based on Chatterjee's and the Keystone team's recommendations, which typically consisted of only one option with no suggested alternatives. *Fourth*, neither Kemerer nor Chatterjee could answer Google's questions about the

1 API package names found in the test scripts. *Fifth*, Google alleges various substantive flaws in
2 Kemerer's analysis, as discussed below.

3 **1. NO PRIOR COLLABORATION.**

4 Google notes that Kemerer and Chatterjee had never worked with each other prior to
5 this case, and that Chatterjee and the Keystone research team have worked with other Oracle
6 experts over the course of this litigation. Google does not articulate what the point of this
7 observation is, or explain why this arrangement between Oracle's experts and Keystone is a
8 basis for exclusion.

9 Nonetheless, Oracle is incorrect in claiming Google's argument is irrelevant to the
10 admissibility of Kemerer's report and testimony. The circumstances of Oracle's collaborations
11 with Keystone at least exacerbate the risk that biased, client-prepared, and litigation-driven tests
12 were spoon-fed to experts to pass off as "facts." *See Therasense, Inc. v. Becton, Dickinson and*
13 *Co.*, Nos. C 04-02123 WHA, C 04-03327 WHA, C 04-03732 WHA, C 05-03117 WHA, 2008
14 WL 2323856, at *2 (N.D. Cal. May 22, 2008). The methodology Kemerer relied on must
15 therefore "be subjected to heightened scrutiny such that it would be reasonable for a truly
16 independent professional in the field of endeavor to base an important decision on it." *Ibid.*

17 **2. THE PYTHON SCRIPTS MISSING FROM KEMERER'S REPORT.**

18 Google complains that although many of the scripts used by Keystone were "provided in
19 hard copy form" with Kemerer's report, certain Python scripts used by Chatterjee as inputs to
20 Kemerer's centrality analysis were not provided with Kemerer's report. Google claims
21 Chatterjee "conceded that the Python scripts were inputs to his use of the Understand and
22 NetworkX tools," and also conceded "that Understand has settings that can be varied," but does
23 not explain the significance of either concession. Moreover, Chatterjee explained in deposition
24 that he used the Python scripts to invoke the Understand and NetworkX tools, and that he used
25 the default settings for the Understand tool (Chatterjee Dep. at 136–41, 154). Google dismisses
26 these explanations, claiming they do not matter because Chatterjee *could have* used other
27 Understand settings, and "Kemerer's description of the use of the Understand program for
28

1 purposes of the ‘centrality’ analysis is so cursory that one could not discern what settings were
2 purportedly used from a review of his report.”

3 Again, it is unclear what Google’s point about admissibility is here. Even if Google
4 could not at first identify the Understand settings used in the centrality analysis based on
5 Kemerer’s report, Google now has that information because Chatterjee explained that he used
6 Understand’s default settings (*id.* at 154). Moreover, as Oracle points out, Google does not
7 claim that any use of these Python scripts or Understand settings undermined the methodology
8 or reliability of Kemerer’s centrality analysis. Kemerer’s report explained that he used
9 Understand to process and analyze Java source files “in order the extract the dependency
10 structure of the app’s source code,” and that “[t]he output of this dependency analysis was
11 subsequently processed” using a Python script “to quantify the app’s dependency on the 37
12 copied Java API packages” (Kemerer Rpt. ¶ 133 & accompanying footnotes). Chatterjee then
13 clarified in deposition that he used Understand’s default settings (Chatterjee Dep. at 154).
14 Nothing in these explanations indicates Kemerer’s methodology was opaque or improper, and
15 Google has shown no reason why Kemerer’s analysis should be excluded as unreliable based on
16 its use of – or his report’s failure to produce – the intermediate Python scripts.

17 **3. ADOPTION OF CHATTERJEE’S RECOMMENDATIONS.**

18 Google also claims that Chatterjee typically gave Kemerer a single option as to which
19 tests to use, without suggesting alternatives, and that Kemerer “approved the Keystone-selected
20 option.” Google’s point seems to be that Kemerer did not actually direct the tests, but merely
21 rubber-stamped the professional judgment of Chatterjee and the Keystone team as to which tests
22 to use. Google cites two examples of this alleged rubber-stamping: the choice to use different
23 methods to gather data from Android and Java, and the choice to use Understand.

24 Google misrepresents Chatterjee’s testimony in claiming that Kemerer did not actually
25 make a “choice” to use different data-collection methods for Android and Java. Chatterjee
26 clearly stated that he did not “make that choice,” but just “pointed [Kemerer] towards what [he]
27 found” in Android’s source code (Chatterjee Dep. at 226). Chatterjee also explained that
28 Kemerer “pointed [him] to the source code,” and that Kemerer “made the choice” to use the

1 particular API documentation files that Chatterjee found (*ibid.*). Kemerer and Chatterjee have
2 consistently claimed that Kemerer directed Chatterjee to find raw data on Android's source
3 code, Chatterjee found Android's API documentation files in plain text and XML, and Kemerer
4 chose to use the files that Chatterjee found (*see* Kemerer Rpt. ¶ 96–99; Chatterjee Dep. at 226).
5 Google has identified no basis for disbelieving this explanation.

6 The real point of Google's argument seems to be that Chatterjee should have found
7 Android's API documentation files in Java instead of plain text and XML; Chatterjee should
8 have offered Kemerer the choice to use the Java files instead; and Kemerer should have used
9 the Java files. The merit of this argument, insofar as it alleges a substantial flaw in Kemerer's
10 methodology, is addressed below. However, to the extent that Google tries to spin this as a
11 rubber-stamping problem by insinuating that Chatterjee, not Kemerer, made the meaningful
12 choice to use plain text and XML files instead of Java files, the argument is meritless. This
13 order disagrees with Google's tortured reasoning that an expert's choice is necessarily "rubber-
14 stamping" unless the expert was explicitly presented with multiple options, even for such
15 mundane "choices" as acceptance of raw data which the expert himself had directed assistants
16 to collect. Chatterjee described finding the raw data that Kemerer had requested, asking
17 Kemerer if the files were "ok" to use, and getting Kemerer's permission to parse the files
18 (Chatterjee Dep. at 166). Under these circumstances, even if Kemerer's choice was not a
19 particularly complex one (and Google has not identified any reason why it should have been),
20 there is no support for the proposition that anyone other than Kemerer — let alone Chatterjee,
21 who acted only according to Kemerer's instructions and with his permission — "directed" the
22 test.

23 Google's second example of alleged rubber-stamping is that "Chatterjee effectively
24 chose" to use Understand without Kemerer's "supervision or substantive technical input."
25 Google cherry-picks from Chatterjee's deposition to suggest Kemerer played no significant part
26 in the decision to use Understand. Reading the relevant part of Chatterjee's deposition as a
27 whole, however, yields a different story. Google asked, "[Y]ou said, Dr. Kemerer, here is the
28 default schema. Is this okay with you? And he said, Yes, it seems fine to me. Is that how it

1 went down?” and Chatterjee replied, “I won’t characterize it exactly that way. He asked me
2 like what’s the — what does Understand exactly capture, I want to know. I gave him the
3 schema. And he — and I asked him, you tell me what you want to pick out of this choice”
4 (Chatterjee Dep. at 155–56). Chatterjee also mentioned that Kemerer asked “a few questions
5 along what this is, what that is” (*id.* at 157). A “couple of days later,” Kemerer told Chatterjee
6 “he was fine using [the Understand schema Chatterjee showed him]” (*id.* at 156). Google also
7 asked, “[D]id you explain to him what is in the schema, or did he just look at the Python file?”
8 and Chatterjee replied, “He took a look at it. He’s done object-oriented programming in the
9 past. So he understands the method class package relationships” (*id.* at 156–57).

10 Chatterjee’s responses show that Kemerer did not perfunctorily approve the use of
11 Understand. If Kemerer had wanted to just rubber-stamp Chatterjee’s suggestion to use the
12 default schema for Understand, he could have simply done so when Chatterjee first suggested it.
13 Instead, Kemerer actively sought to understand the specifics of how Understand worked, asked
14 questions of Chatterjee to that end, reviewed the default schema offered by Chatterjee, and
15 made the ultimate decision to use Understand, apparently after some consideration. This
16 evidence contradicts Google’s claim that Kemerer exercised no supervision or input over the
17 selection of Understand. Furthermore, there is no indication that Chatterjee at any point
18 improperly influenced Kemerer’s decision, or indeed even weighed in on it, aside from
19 providing some answers to Kemerer’s questions about Understand. Thus, there is no support
20 for Google’s assertion that “Chatterjee effectively chose” to use Understand without any
21 supervision or input from Kemerer.

22 4. NUMBER OF API PACKAGES IN THE TEST SCRIPTS.

23 Google next contends that Kemerer’s stability analysis is unreliable because there is no
24 foundation for “the choice to use so many different [API] package names in the scripts.”
25 Google asked both Kemerer and Chatterjee in their respective depositions why the stability
26 analysis references 248 API package names for Java, even though there are only 166 API
27 packages in Java SE 5. Kemerer said he would have to ask his “technical team” (Kemerer Dep.
28 at 178–79), and Chatterjee said Kemerer told him to use those packages (Chatterjee Dep. at

212). Google claims this is “circular finger-pointing” that renders the stability analysis unreliable because “the choice of package names” is unexplained.

This apparent inconsistency arises only if, as Google characterizes it, the use of 248 API package names for Java was a specific choice that someone made. The facts, however, do not support Google’s characterization. When asked in deposition, “So who decided to take these particular 248 APIs?” Chatterjee replied, “It comes from the JDKs that you download from the Oracle website. And so Professor Kemerer told me to do that, and so it comes from there. So he decided” (Chatterjee Dep. at 212). Google then asked, “Who made the decision to use all 248 in this script?” and Chatterjee reiterated, “[Kemerer] made the decision. I mean, he said capture all the packages that are coming from the JDK” (*ibid.*). Google misleadingly abbreviates Chatterjee’s answers as “Kemerer told [Chatterjee] to use those packages,” but omits the rest of Chatterjee’s explanation. Reading Chatterjee’s answers fully and in context, it is clear that the inclusion of 248 API packages was not a specific choice made by either Chatterjee or Kemerer, but simply the result of Kemerer’s earlier instruction to Chatterjee to “capture all the packages that are coming from the JDK.”

Moreover, contrary to Google’s implication, the stability analysis’s inclusion of 248 API packages is not necessarily at odds with the fact that Java SE 5 contains only 166 API packages. As Kemerer’s report explained and Oracle reiterates, the stability analysis considered API packages from all versions of the Java platform, not just from Java SE 5 (Kemerer Rpt. ¶¶ 96–97). Thus, as Oracle puts it, “the total number of packages considered would exceed the package count in any particular version of either platform.” In short, Google has demonstrated neither “circular finger-pointing” nor unreliability in the methodology of Kemerer’s stability analysis.

5. SUBSTANTIVE FLAWS IN KEMERER’S ANALYSIS.

Google alleges three substantial flaws in the methodology of Kemerer’s analysis. For the reasons that follow, this order concludes that all three arguments go to the weight of the analysis, not to its admissibility.

A. Different Data-Gathering Techniques Used for Android and Java.

1 As noted above, Google takes issue with Kemerer's use of plain text and XML, rather
2 than Java, API documentation files in his stability analysis for Android. Specifically, Google
3 "questions . . . whether some of the differences allegedly shown by the analysis could have
4 arisen from the different methods used to gather the underlying data." As a preliminary matter,
5 Oracle correctly points out that all the factual elements underlying this criticism were disclosed
6 in Kemerer's report; it is unclear why this issue was not covered in Google's expert's rebuttal to
7 Kemerer's report, and Google waited until after the Chatterjee deposition to raise it. In any
8 case, Google's objection goes to the weight of Kemerer's analysis, not to its admissibility.
9 Oracle claims it makes less sense to "use an Oracle tool designed for the Java platform to
10 collect information on *Android* source code" than to use "the more direct source of Android
11 method declarations offered by Google," but this, too, is argument by counsel that goes to the
12 weight of expert evidence, not its admissibility. Such arguments must be directed to the jury.

13 **B. Consideration of Non-Java Code in Android.**

14 Android contains both Java and non-Java code, and Chatterjee said in deposition that he
15 thought the centrality analysis could have included both types (Chatterjee Dep. at 238).
16 Kemerer, however, chose to limit the analysis to only Java code in Android (*ibid.*). Google
17 challenges this decision on the basis that "taking into account the entire Android platform could
18 end up presenting a more complete picture of what is supposedly 'central' to the platform."
19 Again, however, this methodological choice and Kemerer's reasoning for it were disclosed in
20 Kemerer's report (Kemerer Rpt. ¶ 150); it is unclear why neither Google nor its experts raised
21 this issue until after Chatterjee's deposition. Additionally, this is another argument that goes to
22 the weight of the analysis, not its admissibility. Even if Google's theory about what a broader
23 analysis of the Android platform might have shown was supported by expert evidence, that
24 evidence would have to be presented to the jury.

25 **C. Design of Stability Analysis.**

26 Finally, Google criticizes Kemerer's stability analysis as "nothing more than a simple
27 counting of changes that have not been shown to make any difference to the platform or to app
28 developers." In other words, the analysis "was not designed to capture changes that would

1 actually be visible to and impact developers,” such as backwards compatibility. Google cites as
 2 support Chatterjee’s admission in deposition that “he did not have expertise in deciding what
 3 sorts of changes would matter to app developers” (Chatterjee Dep. at 60–61). Thus, Google
 4 contends, the stability analysis is “fundamentally unsound” and should be excluded.

5 Google’s argument, however, actually supports Oracle’s position that Kemerer is the
 6 real proffered expert, not a mere mouthpiece, and Chatterjee and the Keystone team are just his
 7 research assistants. Google specifically argues, and argued in prior briefing, that Kemerer
 8 improperly relied on the research team’s opinions or analysis instead of their data collection.
 9 *See Dura Auto. Sys. of Ind., Inc. v. CTS Corp.*, 285 F.3d 609, 614 (7th Cir. 2002). But this
 10 argument is now undermined by Google’s own assertion that Chatterjee’s execution of
 11 Kemerer’s methodology amounted to simple data crunching, not substantive analysis. Such
 12 was the proper role for Chatterjee; he did not need “expertise in deciding what sorts of changes
 13 would matter to app developers” because he was not involved in this case as an expert.
 14 *Kemerer* is the proffered expert who, based on the data crunching performed by Chatterjee and
 15 the Keystone team, purports to opine as to how the results show that stability matters to
 16 developers. Chatterjee’s exercise of professional judgment as to the substantive analysis would
 17 be not only unnecessary, but also problematic under *Dura Auto.* *See* 285 F.3d at 613. After
 18 deposing Chatterjee, Google still has not identified any evidence that Kemerer was a mere
 19 mouthpiece for Chatterjee or any other Keystone expert. At best, Google’s objection amounts
 20 to an attack on the probative value of Kemerer’s stability analysis. As such, however, it would
 21 go to the weight of the analysis, not to its admissibility.

*


*

*

22
 23 The motion to exclude Kemerer’s testimony as having been spoon-fed by Chatterjee is
 24 **DENIED.** Google may use the points developed in the Chatterjee deposition to cross-examine
 25 Kemerer and possibly will be allowed to call Chatterjee in its rebuttal case to impeach Kemerer.

26 **IT IS SO ORDERED.**

27
 28 Dated: May 5, 2016.


 WILLIAM ALSUP
 UNITED STATES DISTRICT JUDGE